Installation Guide to SIMTech_ws

Chen Lequn

December 2019

Contents

1 Ubuntu Installation and python installation

1.1 Install Ubuntu 18.04 on computer

Useful Guides:

`https://blog.csdn.net/Dod_Jdi/article/details/78635126`

`https://blog.csdn.net/heart_1014/article/details/54016687`

`https://help.ubuntu.com/community/DiskSpace`

`https://mp.weixin.qq.com/s/eUF4Hru6PEuZl6Q-ocM0FA`

`https://blog.csdn.net/qq_30638831/article/details/72668821`

`https://www.linuxidc.com/Linux/2018-02/150776.htm`

`https://www.runoob.com/linux/linux-filesystem.html`

`https://blog.csdn.net/wf19930209/article/details/88092752`

`https://zhuanlan.zhihu.com/p/56150978`

`https://blog.csdn.net/qq_39551951/article/details/81093452`

`https://www.zhihu.com/question/39207359`

1.2 Troubleshooting during Ubuntu 18.04 installation

The installation process could vary from different computer, here are some issures may encountered

1.2.1 Important step: partitioning the disk for ubuntu system

Typically, when install ubuntu, the space of each partitions are recommended as below:

- swap (primary partition, begining of space): needs to be twice as physical memory if physical memory is below 5 GB. if not, it is recommended to be 2GB larger than the physical memory. e,g, if you computer memory is 16GB, then swap should be 16-18 GB
- EFI (logical partition, begining of the space): for EFI system partitioning, to boot computer. recommended: 2-4 GB
- Boot: logical, begining of space, Exr4 journal file, mount point on /boot. recommended: 500MB - 1 GB
- root: logical, begining of space, Exr4 journal file, mount on /, recommended: (if you have for example 500G, then give 100 G to this.)
- usr: logical, begining of space, Exr4 journal file, mount on /usr. Normally, should give at least 20G to this. If you have more space, 50-100G recommended.

- home: this works simmilar to 'my computer' in windows – it stores all working space in your PC.logical, begining of space, Exr4 journal file, mount on /home. this partition should be as large as possible. Give all remaining space to this partition

1.2.2 After ubuntu installation. cannot go into ubuntu, enter windows directly

Possible solutions:

- Install EasyBCD software on windows to configure the boot loader program order (but this is only suitable for computer not using EFI boot system (i.e. the legacy support)
- easyEFI software: suitable for most of current computer, configure the grub boot order and let ubuntu to be the first
- when enter the grub, (GUN GRUB) enter 'e' to check, if see 'quiet splash', delete it

1.2.3 Ubuntu system is stucked after boot, or cannot using multiple monitors

This happens when NVIDIA graphical card driver is required.
Solution:

- step 1: go back to grub, tab 'e' and amend the sentece to : "quiet splash nomodeset"
- step 2: after entering the system. Find "software and update", under additional drivers, choose using nvidia driver meta package
- step 3: `https://blog.csdn.net/legalhighhigh/article/details/81448830`

```
    $sudo nano /etc/default/grub
```

  GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
  delete quiet splash, then

```
   $sudo update-grub
```

1.3 Linux Thum drive read only problem:

Linux Thum drive read only problem:
solution:
`https://blog.csdn.net/ustccw/article/details/79040757`

1.4 Install python and python 3 on ubuntu 18.04 (Not Required)

reference: `https://linuxize.com/post/how-to-install-python-3-7-on-ubuntu-18-04/`

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt install python3.7
python3.7 --version
sudo apt install python3-pip
sudo apt install python-pip
```

1.5 Install scikit learn, jupyter etc

everything you did with pip3, please do it again using pip if you are using python2.
Side note: Jupyter lab is **Not Required**

```
pip3 install -U scikit-learn
pip3 install numpy matplotlib
pip3 install scipy
pip3 install jupyterlab
```

1.6 install anaconda on ubuntu (Not Required)

please ignore this section if you are not going to use anaconda environment or jupyter
`https://docs.anaconda.com/anaconda/install/linux/`

```
apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1
libxcomposite1 libasound2 libxi6 libxtst6
```

follow the installation guide from official website, the remeber to activate conda environment by:

```
source ~/anaconda3/bin/activate
```

2 ROS-Industrial Installation

Source code for ROS-industrial:
`https://github.com/ros-industrial`
This is the brief guid on using ROS-industrial to communicate with IRC-5 controller and ABB irb4400 robot.
Prerequisite: ROS "Desktop Full" installed on Ubuntu Linux.
ROS-Industrial stacks enable communication with an industrial robot and various other industrial hardware.
reference: `http://wiki.ros.org/Industrial/Install`
Binary Package Installation:**(note that if you are using ubuntu16.04, you should install kinetic ros version, if you are using ubuntu18.04, you should install melodic version instead)**

- Industrial core:
  for melodic

  ```
  sudo apt-get install ros-melodic-industrial-core
  ```

  for kinetic

  ```
  sudo apt-get install ros-kinetic-industrial-core
  ```

  ———–-IGNORE THE FOLLOWING (NOT RECOMMANDED) —-

- ABB stack: apt-get install ros-kinetic-abb
- Universal robot stack: apt-get install ros-kinetic-universal-robot
- ros_canopen stack: apt-get install ros-kinetic-ros-canopen

2.1 ABB robot packages installation

```
cd ~/ABB_industrial/src
git clone https://github.com/ros-industrial/abb.git
cd ~/ABB_industrial
catkin_make
echo "source ~/ABB_industrial/devel/setup.bash " >> ~/.bashrc
```

3 Install VSCode on Ubuntu 18.04/16.04

reference:
https://linuxize.com/post/how-to-install-visual-studio-code-on-ubuntu-18-04/

- update the packages index and install the dependencies by typing:

  ```
  $ sudo apt update
  $ sudo apt install software-properties-common apt-transport-https wget
  ```

- , import the Microsoft GPG key using the following wget command:

  ```
  $ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
  ```

- enable the Visual Studio Code repository :

  ```
  $ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
  ```

- Once the apt repository is enabled, install the latest version of Visual Studio Code :

  ```
  $ sudo apt update
  $ sudo apt install code
  ```

- Starting Visual Studio Code:
  - launch it either from the command line by typing code
  - or by clicking on the VS Code icon (Activities -¿ Visual Studio Code).

4 Building Qt5 from Source

reference:
`https://blog.csdn.net/friendbkf/article/details/49975775`

4.1 Download qt from official release

- official releas: `http://download.qt.io/official_releases/qt/5.12/5.12.6/`
  (any recent version other than 5.12.6 should also be ok)
  find the file named similar to qt-opensource-linux-x86-5.12.6.run ending with **run** to download

- before installtion, install required librarys as pre-requisite

```
sudo apt-get install build-essential
sudo apt-get install libqt4-dev libgl1-mesa-dev
```

- after downloading, change the .run file executable **please don't directly copy and paste the following command, should use the correct the downloaded file name accordingly**

```
chmod +x qt-opensource-linux-x64-5.12.2.run
```

- double click to install or

```
sudo ./qt-opensource-linux-x64-5.12.6.run
```

5 OpenCV Installation Guide on Ubuntu Linux

5.1 Install OpenCV-Python in Ubuntu

reference:
https://docs.opencv.org/master/d2/de6/tutorial_py_setup_in_ubuntu.html

```
$ sudo apt-get install python-opencv
```

or

```
sudo python -m pip install opencv-python
```

python3:

```
sudo python3 -m pip install opencv-python
```

test code on terminal:

```
import cv2 as cv
print(cv.__version__)
```

5.2 Building OpenCV from source

- https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html
  https://docs.opencv.org/master/d2/de6/tutorial_py_setup_in_ubuntu.html

- The packages can be installed using a terminal and the following commands:

```
[compiler] sudo apt-get install build-essential
[required] sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev
libavformat-dev libswscale-dev
[optional] sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev
libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
```

- Required build dependencies:

```
sudo apt-get install cmake
sudo apt-get install gcc g++
```

```
sudo apt-get install python-dev python-numpy
sudo apt-get install python3-dev python3-numpy
```

we need GTK support for GUI features, Camera support (v4l), Media Support (ffmpeg, gstreamer) etc:

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libgstreamer-plugins-base1.0-dev libgstreamer1.0-dev
```

to support gtk2 and gtk3:

```
sudo apt-get install libgtk2.0-dev
sudo apt-get install libgtk-3-dev
```

optional dependencies:

```
sudo apt-get install libpng-dev
sudo apt-get install libjpeg-dev
sudo apt-get install libopenexr-dev
sudo apt-get install libtiff-dev
sudo apt-get install libwebp-dev
```

- Downloading OpenCV

```
$ sudo apt-get install git
$ git clone https://github.com/opencv/opencv.git
```

important: the downloaded package could be older version of opencv.
therefore, **this method is not recommended**
To install latest **opencv4**: see this link:
`https://blog.csdn.net/new_delete_/article/details/84797041`
Concretely, it is recommended that you should find the official release website here:
`https://opencv.org/releases/`
to download the latest version of opencv

- navigate to the downloaded "opencv" folder. Create a new "build" folder and navigate to it.

```
$ mkdir build
$ cd build
```

- Configuring and Installing:

```
$ cmake ../
$ make
$ sudo make install
```

5.3 ToubleShooting

5.3.1 ImportError: No module named skimage.measure

```
pip install scikit-image
```

As per the official installation page of skimage (skimage Installation) : python-skimage package depends on matplotlib, scipy, pil, numpy.

So install them first using:

```
sudo apt-get install python-matplotlib python-numpy python-pil python-scipy
```

Apparently skimage is a part of Cython which in turn is a superset of python and hence you need to install Cython to be able to use skimage.

```
sudo apt-get install build-essential cython
```

Now install skimage package using:

```
sudo apt-get install python-skimage
```

5.3.2 No module named scipy

```
sudo apt-get install python-scipy
```

or

```
pip install scipy
```

6 Install Point cloud library

- PCL can be installed through PPA (not recommended) you can ignore this PPA installation method

  ```
  sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl
  sudo apt-get update
  sudo apt-get install libpcl1.7 pcl-tools
  ```

  Note that PCL currently supports only C++. There exist a set of Python bindings, but they provide very few functionalities.

  The above instructions will probably not install the latest version of OpenCV and PCL. To get newer version of these libraries, **consider installing from source, which is discribed bellow**

- reference:
  https://www.voidcn.com/article/p-dfdozrzc-bpb.html
  https://machineseez.blogspot.com/2017/06/installing-point-cloud-library-18-on.html
  https://larrylisky.com/2014/03/03/installing-pcl-on-ubuntu/
  https://larrylisky.com/2016/11/03/point-cloud-library-on-ubuntu-16-04-lts/

- Installation from source code process:

  1. Step 1: First things first. Install the dependencies. (Note: includes installation of Boost 1.58)

     ```
     sudo add-apt-repository -y ppa:webupd8team/java

     sudo apt update

     sudo apt -y install oracle-java8-installer

     sudo apt -y install g++ cmake cmake-gui doxygen mpi-default-dev openmpi-bin
     openmpi-common libusb-1.0-0-dev libqhull* libusb-dev libgtest-dev

     sudo apt -y install git-core freeglut3-dev pkg-config build-essential libxmu-dev
     libxi-dev libphonon-dev libphonon-dev phonon-backend-gstreamer

     sudo apt -y install phonon-backend-vlc graphviz mono-complete qt-sdk libflann-dev
     libflann1.8 libboost1.58-all-dev
     ```

     the last one could have some error:

     ```
     E: Unable to locate package qt-sdk
     E: Unable to locate package libflann1.8
     E: Couldn't find any package by glob 'libflann1.8'
     E: Couldn't find any package by regex 'libflann1.8'
     E: Unable to locate package libboost1.58-all-dev
     E: Couldn't find any package by glob 'libboost1.58-all-dev'
     E: Couldn't find any package by regex 'libboost1.58-all-dev'
     ```

     try this instead:

     ```
     sudo apt-get install -y libflann-dev
     sudo apt-get install libboost-all-dev
     ```

  2. Step 2: Install Eigen 3 from Launchpad:
     Use wget or your favorite network downloader:

     ```
     wget http://launchpadlibrarian.net/209530212/libeigen3-dev_3.2.5-4_all.deb
     ```

     Install it using gdebi installer or dpkg on the command line:

     ```
     sudo dpkg -i libeigen3-dev_3.2.5-4_all.deb
     ```

     or :

     ```
     sudo apt-get install libeigen3-dev
     ```

  3. Step 3: Install the latest version of VTK:
     Download:

     ```
     wget http://www.vtk.org/files/release/7.1/VTK-7.1.0.tar.gz
     ```

     decompress:

     ```
     tar -xf VTK-7.1.0.tar.gz
     ```

     Build and install:

     ```
     cd VTK-7.1.0
     mkdir build
     cd build
     cmake ..
     ```

```
make
sudo make install
```

install python vtk

```
python -m pip install vtk
```

4. Step 4: Install PCL 1.8:
   Download and decompress:

   ```
   wget https://github.com/PointCloudLibrary/pcl/archive/pcl-1.8.0.tar.gz
   tar -xf pcl-1.8.0.tar.gz
   ```

   make and build:(This might take a long time)

   ```
   cd pcl-pcl-1.8.0
   mkdir build
   cd build
   cmake ..
   make
   sudo make install
   ```

6.1 Trouble shooting

6.1.1 Solving OpenNI problem: XnOS.h: No such file or directory

reference:
https://programmer.group/solving-openni-problem-xnos.h-no-such-file-or-directory.html

```
$ sudo apt-get install libopenni-dev
```

Install OpenNI2 via apt-get:

```
sudo apt-get update -y
sudo apt-get install -y openni2-dev
```

Install OpenNI2 from source:
reference:
https://github.com/roboticslab-uc3m/installation-guides/blob/master/install-openni-nite.md

```
sudo apt install git libusb-1.0-0-dev libudev-dev
sudo apt install openjdk-8-jdk  # for xenial; openjdk-6-jdk for trusty; if not using other java version.
sudo apt install freeglut3-dev
cd  # go home
mkdir -p repos; cd repos  # create £HOME/repos if it doesn't exist; then, enter it
git clone https://github.com/occipital/OpenNI2.git  # We used to have a fork off 6857677beee08e264fc5aeecb1adf647a7d
cd OpenNI2
make -j$(nproc)  # compile
sudo ln -s $PWD/Bin/x64-Release/libOpenNI2.so /usr/local/lib/  # £PWD should be /yourPathTo/OpenNI2
sudo ln -s $PWD/Bin/x64-Release/OpenNI2/ /usr/local/lib/  # £PWD should be /yourPathTo/OpenNI2
sudo ln -s $PWD/Include /usr/local/include/OpenNI2  # £PWD should be /yourPathTo/OpenNI2
sudo ldconfig
```

7 Micro-epsilon sensor: scanCONTROL

Download from micro-epsilon software:
`https://www.micro-epsilon.com/2D_3D/laser-scanner/Software/downloads/?sLang=en`

- scanCONTROL Linux SDK 0.2.2 (C/C++, Python) (10,6 MB)

7.1 Prerequisite library installation: Aravis

- Aravis is a glib/gobject based library for video acquisition using Genicam cameras. It currently implements the gigabit ethernet and USB3 protocols used by industrial cameras. It also provides a basic ethernet camera simulator and a simple video viewer.

- Installing Aravis: Download the aravis-0.6.1.tar.xz package
  (version 0.7 is not stable, do not download)

  – `http://ftp.acc.umu.se/pub/GNOME/sources/aravis/0.6/`

  After extracting package: Compile

  ```
  ./configure
  sudo make
  sudo make install
  ```

  if encountered problem like: Error Message: configure: error: Your intltool is too old. You need intltool 0.35.0 or later.
  Error Message: checking for GTK... configure: error: Package requirements (gtk+-2.0 ¿= 2.10.0) were not met: No package 'gtk+-2.0' found:

  ```
  $ sudo apt-get update && upgrade
  $ sudo apt-get install intltool
  $ sudo apt-get install libgtk2.0-dev
  ```

- Linux Shared Library Management (**optional read**, no need to take this step because sudo make install already do everything)
  On Ubuntu linux, you may have to configure the **dynamic linker (ld)** to let it know where the aravis libraries are installed, and run ldconfig as root in order to update ld cache:
  - reference on Linux Commands For Shared Library Management & Debugging Problem:
  `https://www.cyberciti.biz/tips/linux-shared-library-management.html`

  Commands

  1. ldconfig : Updates the necessary links for the run time link bindings. create, update, and remove the necessary links and cache (for use by the run-time linker, ld.so) to the most recent shared libraries found in the directories specified on the command line, in the file /etc/ld.so.conf, and in the trusted directories (/usr/lib, /lib64 and /lib).
  2. ldd : Tells what libraries a given program needs to run.
  3. ltrace : A library call tracer.
  4. ld.so/ld-linux.so: Dynamic linker/loader.

  some definitions and usage:

- a **cache** is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere
- Dynamic libraries or linking [also known as DSO (dynamic shared object)] – All lib*.so* files are not copied into executables. The executable will automatically load the libraries using ld.so or ld-linux.so.
- **Static libraries**: – All lib*.a fills are included into executables that use their functions. For example you can run a sendmail binary in chrooted jail using statically liked libs.

The ldconfig commands:

- install the new set of library at:

```
$ ls /home/david/scanControl/aravis-0.6.1
```

- run ldconfig command manually to link libraries by passing them as command line arguments with the -l switch

```
sudo ldconfig -l /path/to/lib/our.new.lib.so
```

- run ldconfig to update the cache:

```
sudo ldconfig
```

- To verify new libs or to look for a linked library, enter:

```
sudo ldconfig -v
sudo ldconfig -v | grep -i geoip
```

7.1.1 Further aravis dependencies are

Compile the basic GNU build toolchain and the autotools are necessary. Further aravis dependencies are:

libxml2 (`https://github.com/GNOME/libxml2/releases`)
glib2 (`https://github.com/GNOME/glib/releases`)

These and the toolchain packages are usually available via the package manager, e.g.

```
$ sudo apt install build-essential autotools-dev automake intltool libtool
$ sudp apt install gtk-doc-tools libxml2-dev libglib2.0-dev
```

7.2 linllt

The linllt libraries (libmescan and libllt) are aravis-based libraries for operating scanCONTROL sensors from MICRO-EPSILON.
libmescan is a C library for sensor control and data aquisition.
linllt is a C++ wrapper.

7.2.1 meson build system: Building the linllt libraries

- the libraries can be built with the [Meson](`http://mesonbuild.com`) build system. Meson checks for the required dependencies. Ninja is used for compiling and installation.
  - meson build system:
  `https://mesonbuild.com/Quick-guide.html`

  install Meson:

  ```
  $ sudo apt install meson
  ```

- For each library in the root folder **(libmescan/ and libllt/)**, execute the following command to compile and install linllt. (libmescan has to be compiled and installed first)

  ```
  $ cd /path to libmescam
  $ meson builddir
  $ cd builddir
  $ ninja
  $ sudo ninja install
  ```

  It might be necessary to reload the linker cache:

  ```
  $ sudo ldconfig
  ```

8 Installation and configuration of Python-pcl library– unsuccessful on Ubuntu18.04 Ignore the whole section 8

8.1 Important note

- This functionality is only tested sucessfully on Ubuntu 16.04 ROS Kinetic version. Due to the python-pcl library is not updatated and is not compatable with ubuntu 18.04, therefore, should not use it when your computer is ubuntu 18.04. We have write the code in C++ with the exact same functionality to replace this node.

- As we use python2 before in all the other ROS program. We should still use the same python version since the python3 and python2 cannot run concurrently in ROS kinetic.
  In the future, it is recommended to build all the python program through python3 because all the new version of ROS will be fully depend on Python3 not Python2.

- sometimes if you use **pip** command to install python library, it may be installed to python3 as default. In order to install the library to Python2, use the command in the format as:

  ```
  $ sudo python2.7 -m install mock
  ```

  note that you may need **sudo** right to execute command, otherwise the system will show permission denied

8.2 Installation

- reference:
  `https://strawlab.github.io/python-pcl`
  `https://blog.csdn.net/zhanghm1995/article/details/86218220`
  `https://github.com/strawlab/python-pcl/issues/71`

- add dependency:

```
$ sudo apt-get install cmake g++ libboost1.58-all-dev libeigen3-dev libflann-dev python
$ sudo apt-get install libusb-1.0{0-dev libudev-dev freeglut3-dev doxygen
$ sudo apt-get install graphviz libpng12-dev libgtest-dev libxmu-dev libxi-dev
$ sudp apt-get install libpcap-dev libqhull-dev libvtk5-qt4-dev python-vtk libvtk-java
```

  p.s. some may not successful, for example python-vtk installation,try:

```
$ python -m pip install vtk
```

- **prerequisite**: install PCL library and build
- add dependent library:

```
pip install --upgrade pip
pip install cython==0.25.2
pip install numpy
```

- installation

```
$ git clone https://github.com/strawlab/python-pcl.git
$ cd python-pcl
$ sudo python setup.py build_ext -i
$ sudo python setup.py install
```

8.3 Troubleshooting

8.3.1 cython error

you may try the following command to install cython 0.26 version

```
$ sudo python -m pip install cython==0.26
```

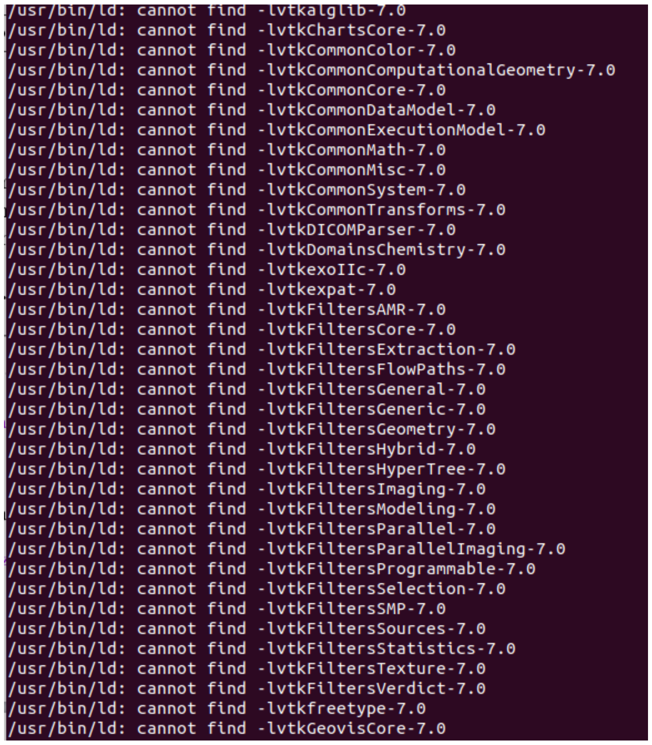
8.3.2 usr/bin/ld: cannot find -lvtkxxx

This problem is cause by a bug in python-pcl.
solution: replace sub-process node written by python-pcl. Rewrite the node in C++.

8.3.3 libpcl_keypoints.so.1.7

ImportError: libpcl_keypoints.so.1.7: cannot open shared object file: No such file or directory
reference:
`https://github.com/strawlab/python-pcl/issues/317`
`https://github.com/strawlab/python-pcl/issues/306`

The reason of this is basically the same as the one in section 7.3.2



```
/usr/bin/ld: cannot find -lvtkalglib-7.0
/usr/bin/ld: cannot find -lvtkChartsCore-7.0
/usr/bin/ld: cannot find -lvtkCommonColor-7.0
/usr/bin/ld: cannot find -lvtkCommonComputationalGeometry-7.0
/usr/bin/ld: cannot find -lvtkCommonCore-7.0
/usr/bin/ld: cannot find -lvtkCommonDataModel-7.0
/usr/bin/ld: cannot find -lvtkCommonExecutionModel-7.0
/usr/bin/ld: cannot find -lvtkCommonMath-7.0
/usr/bin/ld: cannot find -lvtkCommonMisc-7.0
/usr/bin/ld: cannot find -lvtkCommonSystem-7.0
/usr/bin/ld: cannot find -lvtkCommonTransforms-7.0
/usr/bin/ld: cannot find -lvtkDICOMParser-7.0
/usr/bin/ld: cannot find -lvtkDomainsChemistry-7.0
/usr/bin/ld: cannot find -lvtkexoIIc-7.0
/usr/bin/ld: cannot find -lvtkexpat-7.0
/usr/bin/ld: cannot find -lvtkFiltersAMR-7.0
/usr/bin/ld: cannot find -lvtkFiltersCore-7.0
/usr/bin/ld: cannot find -lvtkFiltersExtraction-7.0
/usr/bin/ld: cannot find -lvtkFiltersFlowPaths-7.0
/usr/bin/ld: cannot find -lvtkFiltersGeneral-7.0
/usr/bin/ld: cannot find -lvtkFiltersGeneric-7.0
/usr/bin/ld: cannot find -lvtkFiltersGeometry-7.0
/usr/bin/ld: cannot find -lvtkFiltersHybrid-7.0
/usr/bin/ld: cannot find -lvtkFiltersHyperTree-7.0
/usr/bin/ld: cannot find -lvtkFiltersImaging-7.0
/usr/bin/ld: cannot find -lvtkFiltersModeling-7.0
/usr/bin/ld: cannot find -lvtkFiltersParallel-7.0
/usr/bin/ld: cannot find -lvtkFiltersParallelImaging-7.0
/usr/bin/ld: cannot find -lvtkFiltersProgrammable-7.0
/usr/bin/ld: cannot find -lvtkFiltersSelection-7.0
/usr/bin/ld: cannot find -lvtkFiltersSMP-7.0
/usr/bin/ld: cannot find -lvtkFiltersSources-7.0
/usr/bin/ld: cannot find -lvtkFiltersStatistics-7.0
/usr/bin/ld: cannot find -lvtkFiltersTexture-7.0
/usr/bin/ld: cannot find -lvtkFiltersVerdict-7.0
/usr/bin/ld: cannot find -lvtkfreetype-7.0
/usr/bin/ld: cannot find -lvtkGeovisCore-7.0
```

Figure 1: Error message: usr/bin/ld: cannot find -lxxx

8.4 Test (on terminal)

```
import pcl
import pcl.pcl_visualization

p = pcl.load("/home/david/SIMTech_ws/src/RT_monitoring_application/scanning_robviz/pcl/hammer.pcd")
fil = p.make_statistical_outlier_filter()
fil.set_mean_k(100)
fil.set_std_dev_mul_thresh (1.0)
fil.filter().to_file("/home/david/SIMTech_ws/src/RT_monitoring_application/scanning_robviz/pcl/inliers.pcd")

visual = pcl.pcl_visualization.CloudViewing()
visual.ShowMonochromeCloud(p, b'cloud')
visual.ShowMonochromeCloud(fil.filter(), b'cloud')
```

9 Source Code Download and build for SIMTech_ws

9.1 Source Code Download

- download from github:

```
$ git clone https://github.com/Davidlequnchen/SIMTech_ws.git
```

- build and compile the workspace:

```
$ cd ~/SIMTech_ws
$ catkin_make
```

  refresh the environment(necessary):

```
$ echo "source ~/SIMTech_ws/devel/setup.bash " >> ~/.bashrc
```

  remember should source the .bashrc file in home directory

9.2 ROS perception and USB cam driver installation

- install via apt-get

```
sudo apt-get install ros-melodic-perception
```

  or if using kinetic: **Ignore this step**

```
sudo apt-get install ros-kinetic-preception
```

- Install usb_cam: A ROS Driver for V4L USB Cameras **Ignore this step** (actual usb-cam driver)

```
git clone https://github.com/ros-drivers/usb_cam.git
```

- Install ROS OpenCV camera driver: cv_camera(optional–has not been used in our program) **Ignore this step**

```
git clone https://github.com/OTL/cv_camera.git
```

- Install image_pipline for image processing:(for camera-calibration and undistored image) **Ignore this step**

```
git clone https://github.com/ros-perception/image_pipeline.git
```

- Install Perception-pcl(ros-pcl package for point cloud filtering) **Ignore this step**

```
git clone https://github.com/ros-perception/perception_pcl.git
```

- install pcl_msg:(message files for point cloud usage, optional, not used in our progarm) **Ignore this step**

```
git clone https://github.com/ros-perception/pcl_msgs.git
```

Last but not least: build the package:

```
catkin_make
```

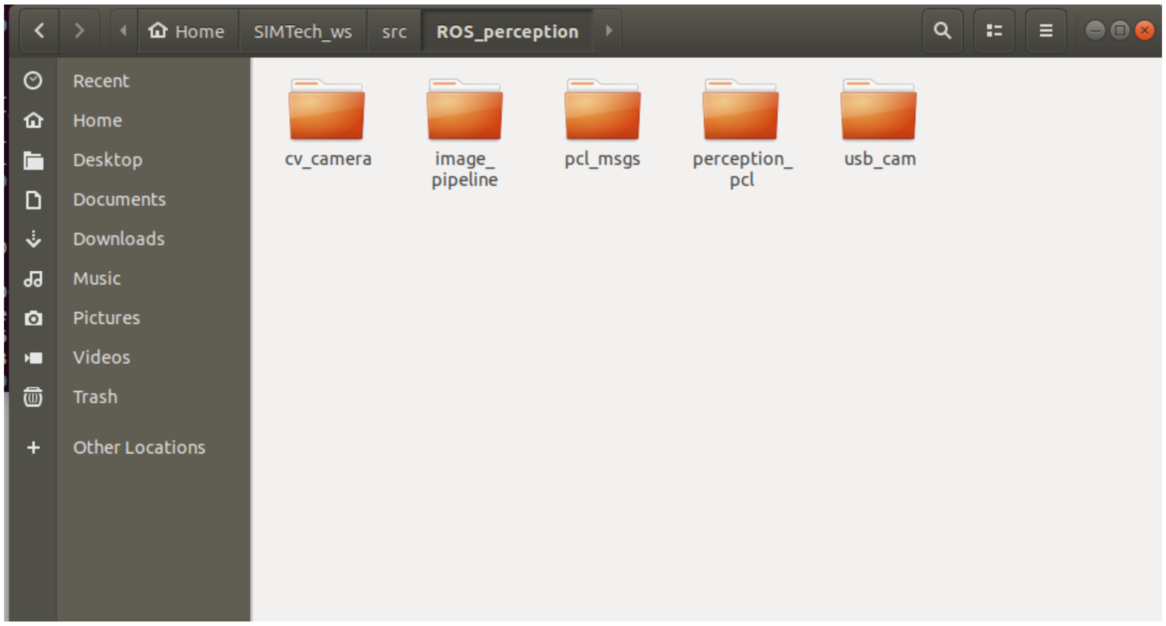


Figure 2: Downloading ros-perception and ros usb-cam driver from source

9.3 Compile the PCL filtering and segmentation program

9.4 PCLFiltering_Tool

- file location:
  /SIMTech_ws/src/scanning_application/Point_cloud_visualziation/PCLFIltering_Tool/build
- Compile and build:

```
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
```

9.5 PCL_segmentation

- file location:
  /SIMTech_ws/src/scanning_application/scanning_robviz/PCL_segmentation/build
- Compile and build:

```
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
```

9.5.1 Troubleshooting: segmentation fault: core dump

solution:
when using cmake to compile the program(**PCL_segmentation.cpp**):
use

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

instead of cmake ..

9.6 Problems troubleshooting

9.6.1 moveit visual tool

Error: Could not find a package configuration file provided by "moveit_visual_tools" with any of the following names: moveit_visual_toolsConfig.cmake moveit_visual_tools-config.cmake
reference:

https://answers.ros.org/question/304475/cannot-build-moveit/
It is stated that you don't have the package moveit_visual_tools installed. You can install it via:

```
sudo apt-get install ros-$ROS_DISTRO-moveit-visual-tools
```

ROS_DISTRO should be replaced by melodic

9.6.2 pcl ros

Ignore this section if no error occurs Add the installation prefix of "pcl_ros" to CMAKE_PREFIX_PATH or set "pcl_ros_DIR" to a directory containing one of the above files. If "pcl_ros" provides a separate development package or SDK, be sure it has been installed.

```
$ sudo apt-get install -y libvtk6-jni libvtk6-java
$ sudo apt-get install -y libvtk6-dev
```

install

```
sudo apt install libpcl-dev ros-melodic-pcl-conversions ros-melodic-pcl-ros
```

9.6.3 usb_cam Error

- reference:
  https://github.com/ros-drivers/usb_cam/issues/100
  https://www.gitmemory.com/issue/ros-drivers/usb_cam/100/491696082
- Problem:

  ```
       (image\787: g\_object\_unref: assertion 'G\_IS\_OBJECT (object)' failed
  ```

  This problem is caused by using older version of ros-perception when you upgrading your system from Ubuntu 16.04 to 18.04 (ROS kinetic to melodic)
- solution: download ros-perception again
- remember to check if you are using the correct usbcam device name (i.e. **/dev/video1**) on different computers, it may be different, for example, it may be **/dev/video2**

10 Install GCC with C++14 support on Ubuntu(If your computer does not have C++14)

reference: http://scholtyssek.org/blog/2015/06/11/install-gcc-with-c14-support-on-ubuntumint/
https://stackoverflow.com/questions/48398475/fail-to-install-gcc-4-9-in-ubuntu17-04

11 Camera calibration file /home/chenlequn/.ros/camera_info/ head_camera.yaml not found

copy the head_camera.yaml file into the folder /.ros/camera_info

- **solution:**

```
cd /home/chenequn/.ros
mkdir camera_info
cp head.yaml ./
```

12 Problem: using python3 and python2 the same time in ROS Melodic Optional read, no need to do this section

12.1 Problem statement:

- As python2 is deprecated since 2020, most of the software packages and python libraries is now only support for python3. Such as pyvrft, point cloud library, pyboof etc.
- However, ROS Melodic does not fully support python3. (The new ROS version **ROS noetic** supports python3 but the it is not mature enough to use).
- Some of the codes (ROS node) is written in python2 and some can only be written in python3 (need to import python3 libararies). Therefore, we need to find a way to make ROS supports both python2 and python3

12.2 Solutions:

- Install the python3 ros packages

```
sudo apt-get install python3-catkin-pkg-modules
```

- Try also:

```
sudo apt-get install python3-catkin-pkg-modules
sudo apt-get install python3-rospkg-modules
```

- then using **catkin_make** to complile the whole catkin workspace. If failed or showed error message like cannot find the packages, then just resintall the ros melodic :

```
sudo apt-get install ros-melodic-desktop-full
```

- When it shows the following error message:
  No module named 'em'
  execute following in the terminal

```
pip3 install empy
```

Note that you must uising pip3 instead of pip2

- Lastly, compile the catkin workspace using:

```
catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3
```

- Side note: do not forget specify the python interpreter at the first line of the python script! This is to specify the path of the interpreter to execute the python code.
  for python2:

  ```
      #!/usr/bin/env python
  ```

  for python3:

  ```
      #!/usr/bin/env python3
  ```

  In this way, both python2 and python3 can work at the same time in the ROS workspace environment.